Discovery and Software Adaptation using a Semantic API

Michael Koster
August 31, 2018

Semantic API

- Enable applications to interact with connected things based on abstract semantics rather than concrete APIs
- Enable discovery and adaptation
- Any application, unmodified, to any connected thing
- Realize the second narrow waist in system design, the complement to common network protocols

Semantic API Examples

```
// Semantic Lookup returns instances capable of semantic lookup
thing = local-directory.lookup-by-simple-template(t);
light = thing( {"@type": ["iot:Light", "BinarySwitchCapability"] } )
switch = light.property( {"@type": "iot:BinarySwitch"} )
rgbcolor = light.property( {"@type": "iot:RGBColor"} )
turnon = light.action( {"@type": "iot:TurnOnAction"} )
setlevel = light.action( {"@type": "iot:SetLevelAction"} )

>>> console.log( switch.read() )
[{ "@type": "iot:BinarySwitchData", "value": true },
  { "@type": "iot:ApplicationTypeData", "value": "unknown" }]

// read() function with DataItem filter
>>> console.log( switch.read( {"@type": "iot:BinarySwitchData"} ))
true

// write() function
switch.write( {"@type": "iot:ApplicationTypeData", "value": "Light"} )
```

Semantic API Examples (2)

```
// Write of multiple DataItems in a structured DataInstance
rgbcolor.write( [
  {"@type": "iot:RedColorData", "value": 255},
  {"@type": "iot:GreenColorData", "value": 255},
  {"@type": "iot:BlueColorData", "value": 255} ] )

// invoke() function
turnon.invoke()

setlevel.invoke( [{"@type": "iot:LevelData", "value": 170},
{"@type": "iot:TransitionTimeData", "value": 100}] )

// chained semantic references
>>> console.log( thing({"@type": ["iot:Light","BinarySwitchCapability"]})
.property({"@type": "iot:BinarySwitch"})
.read({"@type": "iot:BinarySwitchData"}) )
true
```

Semantic Discovery API

- Discover instances of things based on declarative semantics
- Provide for discrete sclectivity and group interaction as appropriate
- Integrate "what to do" and "which one(s)" in the expression

Software Adaptation in WoT

For abstract interaction over diverse ecosystems, adaptation is needed for Transfer Layers, Serialization Formats, and Data Types

1. Adaptation to Transfer Layer formats is provided by Forms element processing
2. **DataInstance class library allows automatic adaptation to diverse serialization formats (OCF, LWM2M, SenML) by embedding a Data Item dictionary that contains Semantic Annotation**
3. **Adaptation to Data Type and Scale + Units may be provided by a DataItem adaptation class and Scale Ontology**

DataInstance Abstract Class

- DataInstance is a representation in some mediatype, which is also a transfer layer payload
- Described by a DataSchema
- Contains one or more DataItem as dataProperty
- Actions and Events exchange DataInstance representations
- Instance of an Interaction Property is an instance of DataInstance
- Interaction Property may also be a instance of a DataItem, providing get() and set() decorators

Semantic annotation and Schemas

- DataItems (variables) in DataInstance Schemas are identified by Semantic Annotation in "@type" values
- Schemas are used to validate and interpret incoming payloads
- Schemas are used to construct outgoing payloads
- Schema validator can be extended to emit a dictionary of DataItems that can be referenced using the Semantic Annotation
- Library can be used to create a Semantic API wrapper for the WoT Scripting API

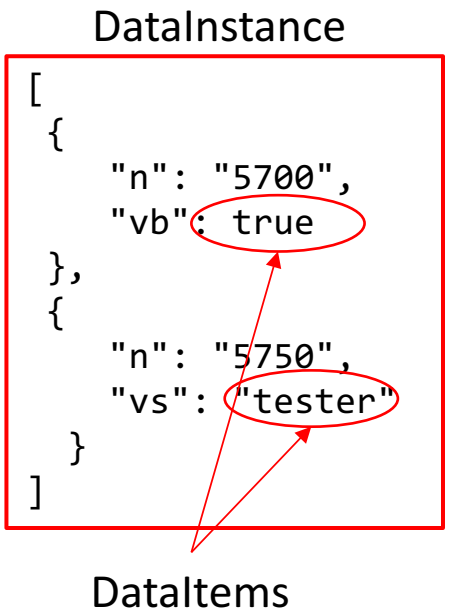
Example Schema with Annotation

```
{
  "type": "array",
  "allOf": [
    {
      "contains": {
        "type": "object",
        "properties": {
          "n": {
            "type": "string",
            "const": "5700"
          },
          "vb": {
            "type": "boolean",
            "@type": "iot:SwitchData"
          }
        },
        "required": ["n", "vb"]
      }
    },
    {
      "contains": {
        "type": "object",
        "properties": {
          "n": {
            "type": "string",
            "const": "5750"
          },
          "vs": {
            "type": "string",
            "@type": "iot:ApplicationTypeData"
          }
        },
        "required": ["n", "vs"]
      }
    }
  ]
}
```

DataInstance Dictionary

- Contains a JSON Pointer and sub-schema for each DataItem in a DataInstance
- Example for a SenML DataInstance

```
[
  {
    "path": "/0/vb",
    "@type": "iot:BinarySwitchData",
    "type": "boolean"
  },
  {
    "path": "/1/vs",
    "@type": "iot:ApplicationTypeData",
    "type": "string"
  }
]
```



Scale and Units Adaptation

- Scale ontology for conversions
- Scale conversions based on slope-intercept form
- Generalize to int-float and coordinate systems
- Tables of Units conversions
- Boundary and exception handling conventions